ELSEVIER
Discrete Applied Mathematics 61 (1995) 187–194
DISCRETE APPLIED MATHEMATICS

Bounds on the size of merging networks

Martin Aigner[a,*], Otfried Schwarzkopf[b]

[a] II. Math. Institute, Freie Universität Berlin, Arnimallee 3, D-14195 Berlin, Germany
[b] Department of Computer Science, Utrecht University, P.O. Box 80089, NL-3508 TB Utrecht, Netherlands

Received 27 October 1992; revised 3 January 1994

Abstract

Let $M(m,n)$ be the minimum number of comparators needed in an (m,n)-merging network. Batcher's odd–even merge provides upper bounds, whereas the best general lower bounds were established by Yao and Yao (1976) and Miltersen et al. (to appear). In this paper, we concentrate on small fixed m and arbitrary n. $M(1,n)=n$ has long been known. In their 1976 paper, Yao and Yao showed $M(2,n)=\lceil 3n/2\rceil$ and asked for the exact value of $M(3,n)$. We prove $M(3,n)=\lceil (7n+3)/4\rceil$ for all n. Furthermore, $M(4,n)>\frac{11}{6}n$, $M(5,n)>2n$ are shown, improving previous bounds. Some related questions are discussed.

1. Introduction

An (m,n)-*merging network* is a sorting network in which the input consists of two sorted sets $\{x_1<\cdots<x_m\}$ and $\{y_1<\cdots<y_n\}$, and the output is the sorted list $\{z_1<z_2<\cdots<z_{m+n}\}$. The components of the network are comparators $i{:}j$, $i<j$, which put $\min(a_i,a_j)$ into position i and $\max(a_i,a_j)$ into position j of the current content $\boldsymbol{a}=(a_1,\ldots,a_{m+n})^{\mathrm{T}}$ (see [4, p. 220]). Let us denote by $M(m,n)$ the minimum number of comparators necessary in the worst case. Batcher's well-known odd–even merge [1; 4, p. 225] provides an upper bound by the recursive inequality

$$M(m,n)\leqslant M(\lceil m/2\rceil,\lceil n/2\rceil)+M(\lfloor n/2\rfloor,\lfloor n/2\rfloor)+\lfloor m/2\rfloor+\lfloor n/2\rfloor, \tag{1}$$

where the right-hand side is lowered by 1 if both m and n are even. Asymptotically, this gives

$$M(m,n)\leqslant \tfrac{1}{2}(m+n)\log_2(m+1)+\mathrm{O}(n)\quad (m\leqslant n). \tag{2}$$

Floyd [3] proved the following recursive inequality for lower bounds:

$$M(m,n)\geqslant M(m_1,n_1)+M(m_2,n_2)+\max(\min(m_1,n_2),\min(m_2,n_1))$$
$$\text{for } m=m_1+m_2,\ n=n_1+n_2, \tag{3}$$

* Corresponding author.

SSDI 0166-218X(94)00015-6

which for $m = n$ yields

$$M(n,n) \geqslant \tfrac{1}{2} n \log_2 n + \mathrm{O}(n). \tag{4}$$

For general m and n, Yao and Yao [6] showed

$$M(m,n) \geqslant \frac{\log_2(m+1)}{2} n \quad (m \leqslant n), \tag{5}$$

and, quite recently, Miltersen et al. [5] improved this last result to

$$M(m,n) \geqslant \tfrac{1}{2}(m+n)\log_2(m+1) - \mathrm{O}(m), \tag{6}$$

which asymptotically matches Batcher's upper bound (2).

In this paper we address the question of obtaining better lower bounds for $M(m,n)$ for small fixed m and arbitrary n. Note that for fixed m, (6) is not a real improvement over (5). It has long been known that $M(1,n) = n$, Yao and Yao proved $M(2,n) = \lceil 3n/2 \rceil$ for all n in their paper [6]. In that same paper they raised the question as to the exact value of $M(3,n)$. Our main result gives the precise answer

$$M(3,n) = \left\lceil \frac{7n+3}{4} \right\rceil.$$

Furthermore, we derive the bounds $M(4,n) \geqslant \frac{11}{6}n + 1$, $M(5,n) \geqslant 2n + 1$, and make some comments on other values $M(m,n)$ for small m and n. Note that (1) and (5) yield $\frac{1}{2}(\log_2 5)n \leqslant M(4,n) \leqslant 2n + 2$ and $\frac{1}{2}(\log_2 6)n \leqslant M(5,n) \leqslant \frac{17}{8}n + C$.

2. Preliminary results

As usual, we picture our network as $m + n$ horizontal lines numbered from top to bottom, where the inputs of the first m and last n lines are sorted, with the smallest element appearing on top of the lists. The output of the merging network is sorted in ascending order from top to bottom.

Denote by $\boldsymbol{n}$ the vector $\boldsymbol{n} = (1, 2, \ldots, n)^{\mathrm{T}}$, and by $\boldsymbol{b}_i$ the vector $\boldsymbol{b}_i = (-(m-1-i), \ldots, -1, 0, n+1, \ldots, n+i)^{\mathrm{T}}$, $i = 1, \ldots, m$, with $\boldsymbol{b}_m = (n+1, \ldots, n+m)^{\mathrm{T}}$. Finally, let $\boldsymbol{a}_i$ be the vector $\boldsymbol{a}_i = (\boldsymbol{b}_i / \boldsymbol{n})$, $i = 1, \ldots, m$. As an example, for $m = 3$, we obtain

$$\boldsymbol{a}_1 = \begin{pmatrix} -1 \\ 0 \\ n+1 \\ 1 \\ \vdots \\ n \end{pmatrix}, \quad \boldsymbol{a}_2 = \begin{pmatrix} 0 \\ n+1 \\ n+2 \\ 1 \\ \vdots \\ n \end{pmatrix}, \quad \boldsymbol{a}_3 = \begin{pmatrix} n+1 \\ n+2 \\ n+3 \\ 1 \\ \vdots \\ n \end{pmatrix}.$$

Lemma 1. *If an (m,n)-merging network sorts $\boldsymbol{a}_1,\dots,\boldsymbol{a}_m$, then it sorts any m-lists and n-lists.*

Proof. By the 0, 1-principle (see [4, p.224]) we have to show that if an (m,n)-merging network sorts $\boldsymbol{a}_1,\dots,\boldsymbol{a}_m$ then it sorts any m-lists and n-lists consisting of 0's and 1's. The sorted 0, 1-lists of length m are $\boldsymbol{c}_0,\boldsymbol{c}_1,\dots,\boldsymbol{c}_m$ with

$$\boldsymbol{c}_i = (0,\dots,0,\underbrace{1,\dots,1}_{i})^{\mathrm{T}}$$

where $\boldsymbol{c}_0$ is of no relevance since in this case the lines are already sorted. Suppose, on the contrary, that $\boldsymbol{x} = (\boldsymbol{c}_i/\boldsymbol{d})$ is not properly sorted with $y_k = 1$, $y_{k+1} = 0$ in the output $\boldsymbol{y}$. Let h be the largest index in $\boldsymbol{d}$ with $d_h = 0$ (where we set $h = 0$ if all $d_j = 1$). We define $f: \boldsymbol{x} \to \boldsymbol{a}_i$ coordinatewise. Since f is monotone on the x_i's, the input $f(\boldsymbol{x}) = \boldsymbol{a}_i$ is carried into $f(\boldsymbol{y})$ which is sorted by assumption. However, from $y_k = 1$ we infer $f(y_k) \geqslant h+1$ and from $y_{k+1} = 0$ we have $f(y_{k+1}) \leqslant h$, in contradiction to $f(y_k) < f(y_{k+1})$. $\square$

We will make use of another well-known result ([4, p. 239, Ex. 21]).

Lemma 2. *Let π be an (m,n)-merging network. If two inputs $\boldsymbol{a},\boldsymbol{a}'$ satisfy $\boldsymbol{a} \leqslant \boldsymbol{a}'$ (i.e. $a_i \leqslant a_i'$ for all i), then at any stage l of π we have $\boldsymbol{a}^{(l)} \leqslant \boldsymbol{a}'^{(l)}$. In particular, $\pi\boldsymbol{a} \leqslant \pi\boldsymbol{a}'$ holds for the respective outputs.*

Notice that our vectors $\boldsymbol{a}_i$ satisfy $\boldsymbol{a}_1 \leqslant \boldsymbol{a}_2 \leqslant \cdots \leqslant \boldsymbol{a}_m$.

3. The case $M(1,n)$

Let us analyze $(1,n)$-networks since our later proofs hinge on this case. Lemma 1 tells us that we may confine our attention to the single input $\boldsymbol{a} = (n+1, 1,\dots,n)^{\mathrm{T}}$. We can clearly sort $\boldsymbol{a}$ with n comparisons by moving $n+1$ down step by step. Let us analyze an arbitrary network sorting $\boldsymbol{a}$. We claim that the content $a_1,\dots,a_{n+1}$ at a certain stage can be described as follows. The lines are uniquely divided into groups $F_1|F_2|\cdots|F_r$ from top to bottom such that

(i) $i \in F_k, j \in F_l, k < l \Rightarrow a_i < a_j$.

(ii) if $F_k = \{f_k, f_k+1,\dots,f_{k+1}-1\}$, then $a_{f_k+1} < \cdots < a_{f_{k+1}-1} < a_{f_k}$.

In other words, within each group the top element is the largest, with the others appearing in their natural order.

To see this, note that at the start we have a single group $F_1 = \{n+1, 1,\dots,n\}$ satisfying (ii). Now assume that we compare at a certain stage lines i and j, $i < j$. If i and j are in different groups, then nothing changes by (i). If i and j are in group F_k with $f_k < i < j \leqslant f_{k+1}-1$, then again nothing happens by (ii). If finally i and j are in F_k with $i = f_k < j \leqslant f_{k+1}-1$, then the contents of lines i and j are interchanged, and F_k splits into two groups $F_k' = \{j, f_k+1,\dots,j-1\}$, $F_k'' = \{i, j+1,\dots,f_{k+1}-1\}$. With F_k' and F_k'' replacing F_k, we see that (i) and (ii) are again satisfied.

We conclude that any comparison raises the number of groups by at most 1. Since we have $n+1$ groups $\{1\},\{2\},\ldots,\{n+1\}$ at the end, we infer that n comparisons are needed to sort $\boldsymbol{a}$. Thus $M(1,n)=n$.

Note that we have proved that in any algorithm sorting $\boldsymbol{a}$ there are *precisely* n comparisons which change the current content. Let us call them *essential comparisons*. Note also that after the first essential comparison $i:j$, $i<j$, involving line j, the content of line j moves to the top of that group and stays at the top of its respective group thereafter. Hence, after an essential comparison $i:j$, $i<j$, the content of line j can only be changed by comparisons $j:k$, $j<k$.

4. Proof of $M(3,n)=\lceil(7n+3)/4\rceil$

Using (1) and the result $M(2,n)=\lceil 3n/2\rceil$, it is readily established that Batcher's odd–even merge for $(3,n)$-networks uses $\lceil(7n+3)/4\rceil$ comparisons. To verify $M(3,n)\geqslant\lceil(7n+3)/4\rceil$ we appeal to Lemma 1. We have to sort $\boldsymbol{a}=(-1,0,n+1,1,\ldots,n)^{\mathrm{T}}$, $\boldsymbol{b}=(0,n+1,n+2,1,\ldots,n)^{\mathrm{T}}$, $\boldsymbol{c}=(n+1,n+2,n+3,1,\ldots,n)^{\mathrm{T}}$, where $\boldsymbol{a}\leqslant\boldsymbol{b}\leqslant\boldsymbol{c}$. Let us number the lines $-2,-1,0,1,\ldots,n$ from top to bottom.

Suppose an optimal network π needs l comparisons. Let $(a_{-2},a_{-1},a_0,\ldots,a_n)$, $(b_{-2},\ldots,b_n)$, $(c_{-2},\ldots,c_n)$ be the contents of $\boldsymbol{a},\boldsymbol{b},\boldsymbol{c}$ after the hth comparison. From the input $\boldsymbol{a},\boldsymbol{b},\boldsymbol{c}$ at the outset and Lemma 2 it is clear that

$$c_j\geqslant b_j\geqslant a_j\geqslant j\quad\text{for } j=1,\ldots,n. \tag{7}$$

We set

$$d_h=|\{i\colon a_i\neq b_i\}|+|\{i\colon a_i\neq c_i\}|+|\{i\colon b_i\neq c_i\}|. \tag{8}$$

Thus, at the outset $d_0=9$, and at the end $d_l=3n+9$, hence $d_l-d_0=3n$. Since the top two entries of $\boldsymbol{a}$ will never be moved, the action of π on $\boldsymbol{a}$ will be that of the previous case $M(1,n)$. We know these are precisely n essential comparisons which change the content of $\boldsymbol{a}$; let us call them *inner* comparisons of π, with the others being called *outer* comparisons.

Suppose the $(h+1)$st comparison is $i:j$, $i<j$. It is readily seen that $d_{h+1}-d_h\leqslant 2$. An inner comparison which raises d_h is called a *red* comparison; let r be the number of red comparisons. Let us analyze a red comparison $i:j$, $i<j$. Since it is an essential comparison for $\boldsymbol{a}$ we must have $a_i>a_j$, and thus $j\geqslant 1$. If $a_j=b_j=c_j$, then by Lemma 2, $c_i\geqslant b_i\geqslant a_i>a_j=b_j=c_j$ implying $d_{h+1}=d_h$. Thus, for a red comparison we must have $a_j<c_j$. Now, by the remark at the end of the last section, line j was not involved in an inner comparison before. Since the initial content of line $j\geqslant 1$ is (j,j,j), it must have been part of a previous outer comparison. Note that the first comparison changing the content (j,j,j) is of the form $i':j$, $i'<j$ by (7).

Hence, we conclude that there are at least r outer comparisons, i.e. $l-n\geqslant r$. We are going to show that, in fact, $l-n\geqslant r+1$. To see this, consider the first comparison $-1:j$, involving line -1, with the contents $(0,n+1,n+2)$. This is at any rate an outer comparison, and it is an additional outer comparison unless $j \geqslant 1$, and $-1:j$ is also the first comparison involving line j. After the comparison, the content of line -1 is $(0, j, j)$. Hence, there must be another, again outer comparison $-1:k$, in order to "break" $j:j$. Suppose the content of line k is (u, v, w). In order to break j, j we must have $v < j < w$, in particular $u < w$, which means that $-1:k$ is an additional outer comparison not yet accounted for.

From $d_l - d_0 = 3n$ and $d_{h+1} \leqslant d_h + 1$, we infer $2(l - n + r) \geqslant 3n$, and hence by $l - n \geqslant r + 1$,

$$3n \leqslant 2(l - n + r) \leqslant 2(l - n) + 2(l - n - 1) = 4l - 4n - 2,$$

thus

$$l \geqslant \frac{7n + 2}{4}.$$

For $n \not\equiv 2 \pmod 4$, we have $\lceil (7n + 2)/4 \rceil = \lceil (7n + 3)/4 \rceil$, so for $n \not\equiv 2 \pmod 4$, our theorem is proved. In the case $n \equiv 2 \pmod 4$ we only have $\lceil (7n + 2)/4 \rceil = \lceil (7n + 3)/4 \rceil - 1$. A more detailed analysis (considering the first comparison involving line -2) shows that in this case one more comparison is indeed necessary, thereby completing the proof of the theorem.

5. The cases $M(4, n)$ and $M(5, n)$

Using the result $M(2, n) = \lceil 3n/2 \rceil$ and (1), one computes that Batcher's odd–even merge uses $2n + 1$ comparisons for $n \equiv 0 \pmod 4$ and $2n + 2$ for $n \not\equiv 0 \pmod 4$. By Lemma 1, we must sort the vectors $\boldsymbol{a} = (-2, -1, 0, n + 1, 1, \ldots, n)^{\mathrm{T}}$, $\boldsymbol{b} = (-1, 0, n + 1, n + 2, 1 \ldots, n)^{\mathrm{T}}$, $\boldsymbol{c} = (0, n + 1, n + 2, n + 3, 1, \ldots, n)^{\mathrm{T}}$, $\boldsymbol{d} = (n + 1, n + 2, n + 3, n + 4, 1, \ldots, n)^{\mathrm{T}}$ with $\boldsymbol{a} \leqslant \boldsymbol{b} \leqslant \boldsymbol{c} \leqslant \boldsymbol{d}$. All the notions of inner, outer and red comparison carry over, since the action on $\boldsymbol{a}$ is that of the $M(1, n)$-case. Hence, we infer again $l - n \geqslant r$, and in fact, $l - n \geqslant r + 2$. The proof of the $M(3, n)$-case was based on the fact that $d_l - d_0 = 3n$, $d_{h+1} \leqslant d_h + 2$. So we must find a *weighting* w of the content (a, b, c, d) of a line analogous to (8), with $d_h = \sum_i w(a_i, b_i, c_i, d_i)$.

There are four possibilities according to the number of equality signs between a, b, c, d. Let us choose weights $0 = \alpha_0 < \alpha_1 < \alpha_2 < \alpha_3$ as follows:

$$\begin{array}{llll} 0 = \alpha_0: & a = b = c = d, & \alpha_2: & a = b < c < d, \\ \alpha_1: & a = b = c < d, & & a < b = c < d, \\ & a = b < c = d, & & a < b < c = d, \\ & a < b = c = d. & \alpha_3: & a < b < c < d. \end{array}$$

Thus at the outset $d_0 = 4\alpha_3$ and at the end $d_l = (4 + n)\alpha_3$, hence $d_l - d_0 = \alpha_3 n$. Any $x > 0$ satisfying $d_{h+1} - d_h \leqslant x\alpha_3$ for all h will then provide a lower bound for l by the same argument as in the $M(3,n)$-case. Indeed, from $r \leqslant l - n - 2$ we infer

$$\alpha_3 n \leqslant x\alpha_3(l - n + r) \leqslant 2x\alpha_3(l - n - 1),$$

and thus

$$l \geqslant \left(1 + \frac{1}{2x}\right)n + 1. \tag{9}$$

To obtain the best bound in (9) we must find the smallest $x > 0$ satisfying $d_{h+1} - d_h \leqslant x\alpha_3$. Analyzing the possible changes $d_h \to d_{h+1}$ it is readily seen that under the assumption $0 = \alpha_0 < \alpha_1 < \alpha_2 < \alpha_3$, the largest possible increases $d_{h+1} - d_h$ are $2\alpha_2 - \alpha_3$, α_1, $2\alpha_3 - \alpha_1 - \alpha_2$, $2\alpha_2 - 2\alpha_1$. Solving the inequalities $2\alpha_2 - \alpha_3 \leqslant x\alpha_3$, $\alpha_1 \leqslant x\alpha_3$, $2\alpha_3 - \alpha_1 - \alpha_2 \leqslant x\alpha_3$, $2\alpha_2 - 2\alpha_1 \leqslant x\alpha_3$ yields as minimal value $x = \frac{3}{5}$ with a realization $\alpha_0 = 0$, $\alpha_1 = 3$, $\alpha_2 = 4$, $\alpha_3 = 5$. Hence, we conclude from (9) that

$$M(4,n) \geqslant \tfrac{11}{6}n + 1.$$

Let us now look at $M(5,n)$. Here we must choose the weighting a little more carefully. We choose weights $0 = \alpha_0 < \alpha_1 < \alpha_2 < \alpha_2' < \alpha_3 < \alpha_3' < \alpha_4$ as follows:

$$\begin{array}{llll}
0 = \alpha_0: & a = b = c = d = e, & \alpha_2': & a = b < c = d < e, \\
 & & & a = b < c < d = e, \\
\alpha_1: & a = b = c = d < e, & & a < b = c = d < e, \\
 & a = b = c < d = e, & & a < b = c < d = e, \\
 & a = b < c = d = e, & & \\
 & a < b = c = d = e, & \alpha_3: & a = b < c < d < e, \\
 & & & a < b < c < d = e, \\
\alpha_2: & a = b = c < d < e, & & \\
 & a < b < c = d = e, & \alpha_3': & a < b = c < d < e, \\
 & & & a < b < c = d < e, \\
 & & \alpha_4: & a < b < c < d < e.
\end{array}$$

One now finds that the largest possible increases of $d_{h+1} - d_h$ are

$$\begin{array}{ll}
\alpha_1, & 2\alpha_3' - \alpha_1 - \alpha_2', \\
\alpha_2 + \alpha_2' - \alpha_3', & \alpha_3 + \alpha_4 - \alpha_1 - \alpha_3', \\
\alpha_2 + \alpha_3 - \alpha_4, & \alpha_2' + \alpha_4 - 2\alpha_2, \\
2\alpha_2' - 2\alpha_1, & 2\alpha_3' - 2\alpha_2, \\
\alpha_2' + \alpha_3' - \alpha_1 - \alpha_2, & \alpha_3' + \alpha_4 - \alpha_2 - \alpha_3, \\
\alpha_2 + \alpha_4 - \alpha_1 - \alpha_3, & 2\alpha_4 - 2\alpha_2'.
\end{array}$$

Solving the 12 inequalities $\alpha_1 \leqslant x\alpha_4, \ldots, 2\alpha_4 - 2\alpha_2' \leqslant x\alpha_4$ yields as minimal value $x = \frac{1}{2}$ with a realization $\alpha_0 = 0$, $\alpha_1 = 8$, $\alpha_2 = 10$, $\alpha_2' = 12$, $\alpha_3 = 13$, $\alpha_3' = 14$, $\alpha_4 = 16$. Hence, we obtain as in (9)

$$M(5, n) \geqslant 2n + 1.$$

6. Some exact values and open questions

The main theorem can be used to derive some exact results for $M(n, n)$ for small n. The upper bounds follow from Batcher's merge (1), the lower bounds from Floyd's [3] result.

Corollary. *We have* $M(2,2) = 3$, $M(3,3) = 6$, $M(4,4) = 9$, $M(5,5) = 13$, $M(7,7) = 21$, $M(8,8) = 25$, $M(9,9) = 30$.

As an example, consider $M(9,9)$. By (3) and our theorem, $M(9,9) \geqslant M(3,6) + M(6,3) + 6 = 12 + 12 + 6 = 30$. For $M(6,6)$, odd–even merge yields $M(6,6) \leqslant 17$, but the best we can do applying (3) is $M(6,6) \geqslant M(2,4) + M(4,2) + 4 = 16$.

Batcher's odd–even merge is thus optimal for $M(m, n)$, $m \leqslant 3$ and all n, and for some $M(n, n)$, and it seems plausible that it is indeed always optimal.

Problem 1. Is Batcher's odd–even merge optimal for all m and n?

The available data suggest several recursive inequalities for the numbers $M(m, n)$. For example, it is immediately clear that $M(m, n) \geqslant M(m-1, n) + 1$.

Problem 2. Is $M(m, n) \geqslant M(m-1, n) + 2$ for all $m < n$?

Since $M(3,6) = 12$, this would e.g. imply $M(4,6) = 14$, $M(5,6) = 16$ and $M(6,6) = 17$. Table 1 gives the best lower and upper bounds for $M(m, n)$ with $m, n \leqslant 9$.

Table 1

m	n							
	2	3	4	5	6	7	8	9
2	3	5	6	8	9	11	12	14
3		6	8	10	12	13	15	17
4			9	11/12	13/14	14/16	16/17	18/20
5				13	14/16	16/18	18/20	20/22
6					16/17	18/20	20/22	21/25
7						21	23/24	24/27
8							25	27/29
9								30

References

[1] K.E. Batcher, Sorting networks and their applications, in: Proceedings AFIPS 1968 S3CC, Vol. 32 (AFIPS Press, Montvale, 1968) 307–314.
[2] M. Chung and B. Ravikumar, On the size of test sets for sorting and related problems, in: Proceedings 1987 International Conference on Parallel Processing (Penn State Press, 1987).
[3] R.W. Floyd, Permuting information in idealized two-level storage, in: R.E. Miller and J.W. Thatcher, eds., Complexity of Computer Computations (Plenum Press, New York, 1972) 105–110.
[4] D.E. Knuth, The Art of Computer Programming, Vol. 3 (Addison-Wesley, Reading, MA, 1973).
[5] P.B. Miltersen, M. Paterson and J. Tarui, The asymptotic complexity of merging networks, in: FOCS'92, to appear.
[6] A.C.C. Yao and F.F. Yao, Lower bounds on merging networks, J. ACM 23 (1976) 566–571.